# Derandomizing pseudo-random sequences using MapReduce on Amazon Web Services

**Cătălin-Alexandru Avram**

March 22nd 2010

# Overview

- Introduction
- Pseudo-random number generators
- Stream ciphers
- Amazon elastic map reduce
- Our solution & Experiments
- Conclusion
- Questions

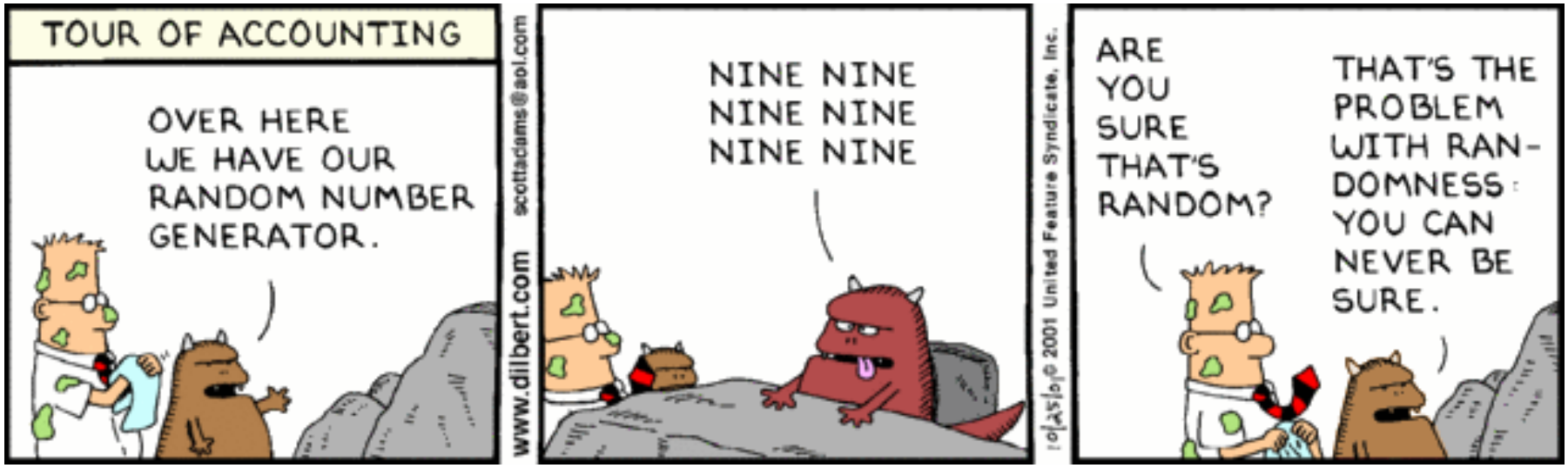
# Introduction

- This is an experiment type project
- The subject of the experiment is Amazon's Elastic MapReduce web service
- The algorithm used for testing finds the initial seed of a given pseudo-random sequence
- Our main interests are the web service's usability, price and performance

# Pseudo-random number generators

- Don’t produce real randomness
- Work by applying a deterministic mathematical function to the previously generated value
- Produce sufficiently uniform distributions to be used in statistical simulations
- Used in cryptography (as inputs for stream ciphers)
- Susceptible to brute force attacks on the initial seed

# Pseudo-random number generators



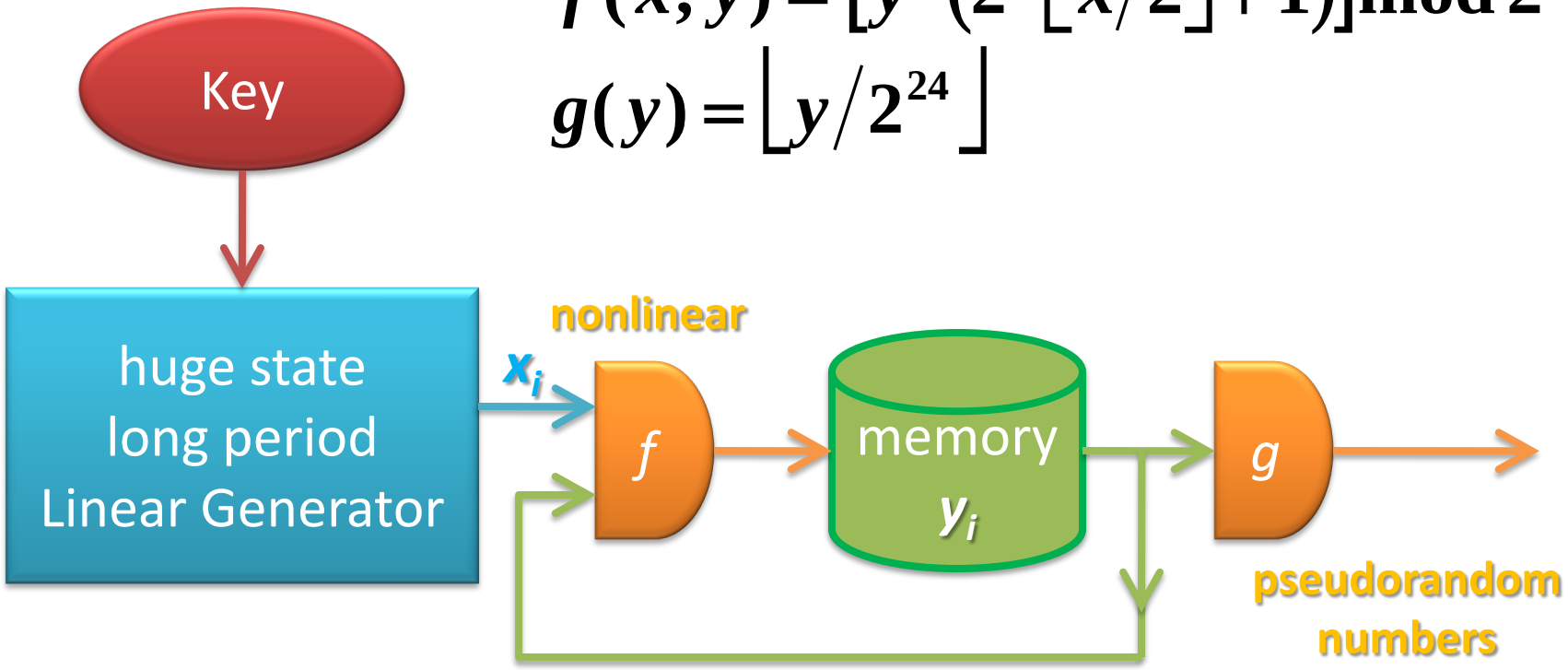
# Stream ciphers

- "In cryptography, a stream cipher is a symmetric key cipher where plaintext bits are combined with a pseudorandom cipher bit stream (keystream).
- In a stream cipher the plaintext digits are encrypted one at a time.
- An alternative name is a state cipher, as the encryption of each digit is dependent on the current state." - Wikipedia

# CryptMT

$$f(x,y) = \left[ y \cdot \left( 2 \cdot \lfloor x/2 \rfloor + 1 \right) \right] \bmod 2^{32}$$

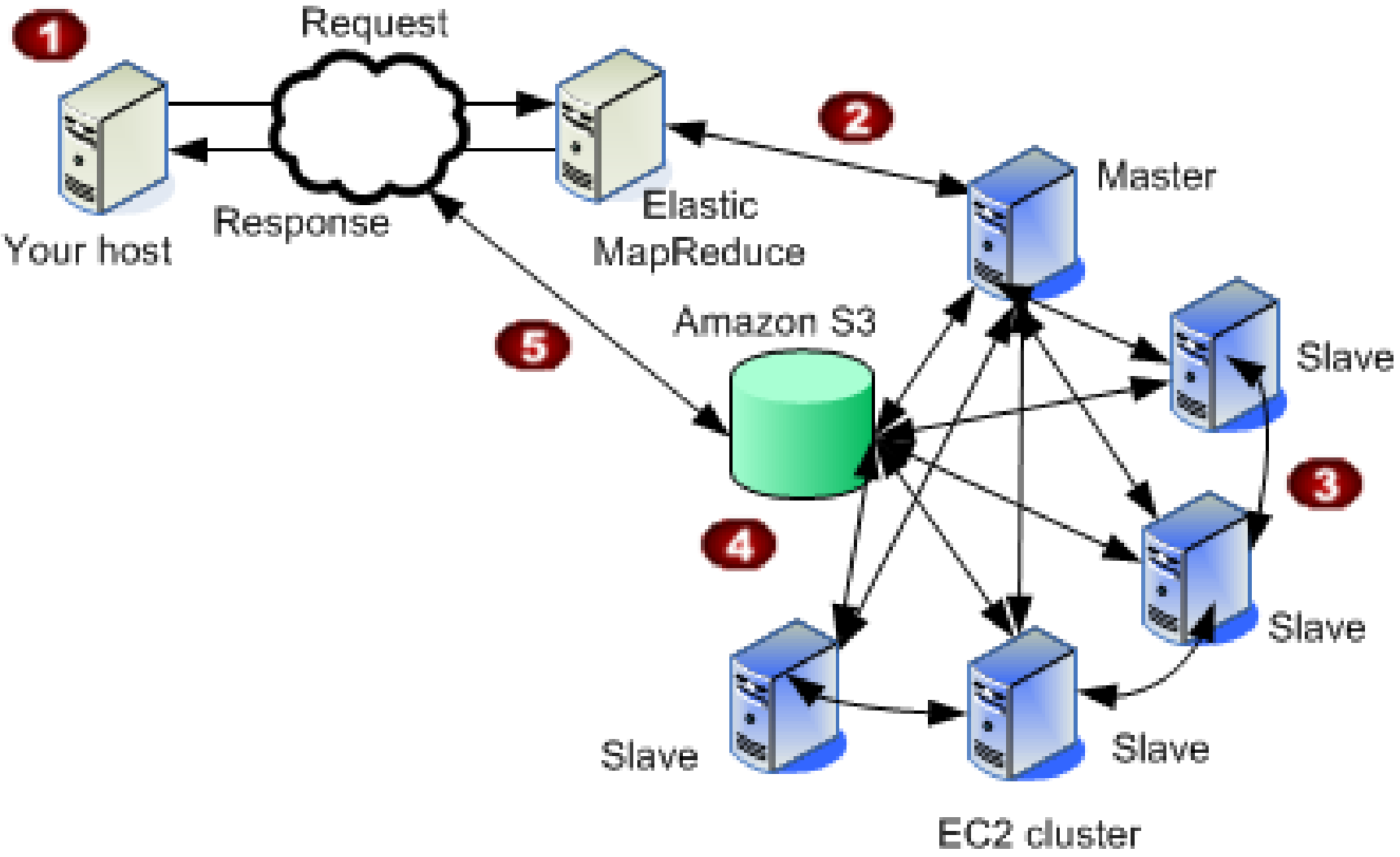
$$g(y) = \lfloor y / 2^{24} \rfloor$$



# Mersenne twister

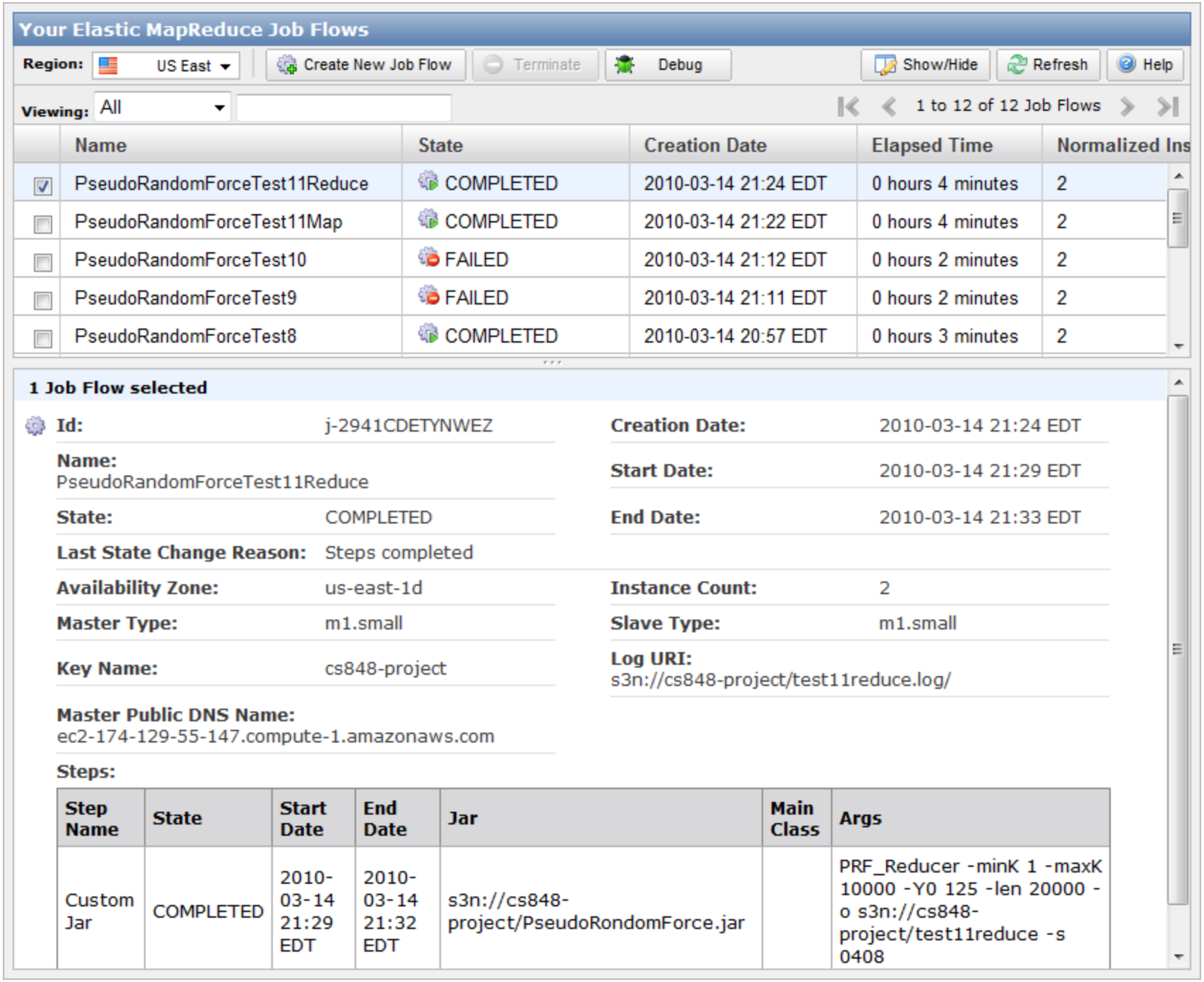
- Period of $2^{19937} - 1$
- 623 dimensional equidistribution
- 32 bit accuracy
- 624 words of working memory required in implementation
- Observing just 624 consecutive generated numbers allows one to compute the entire internal state of the generator

# Amazon Elastic MapReduce

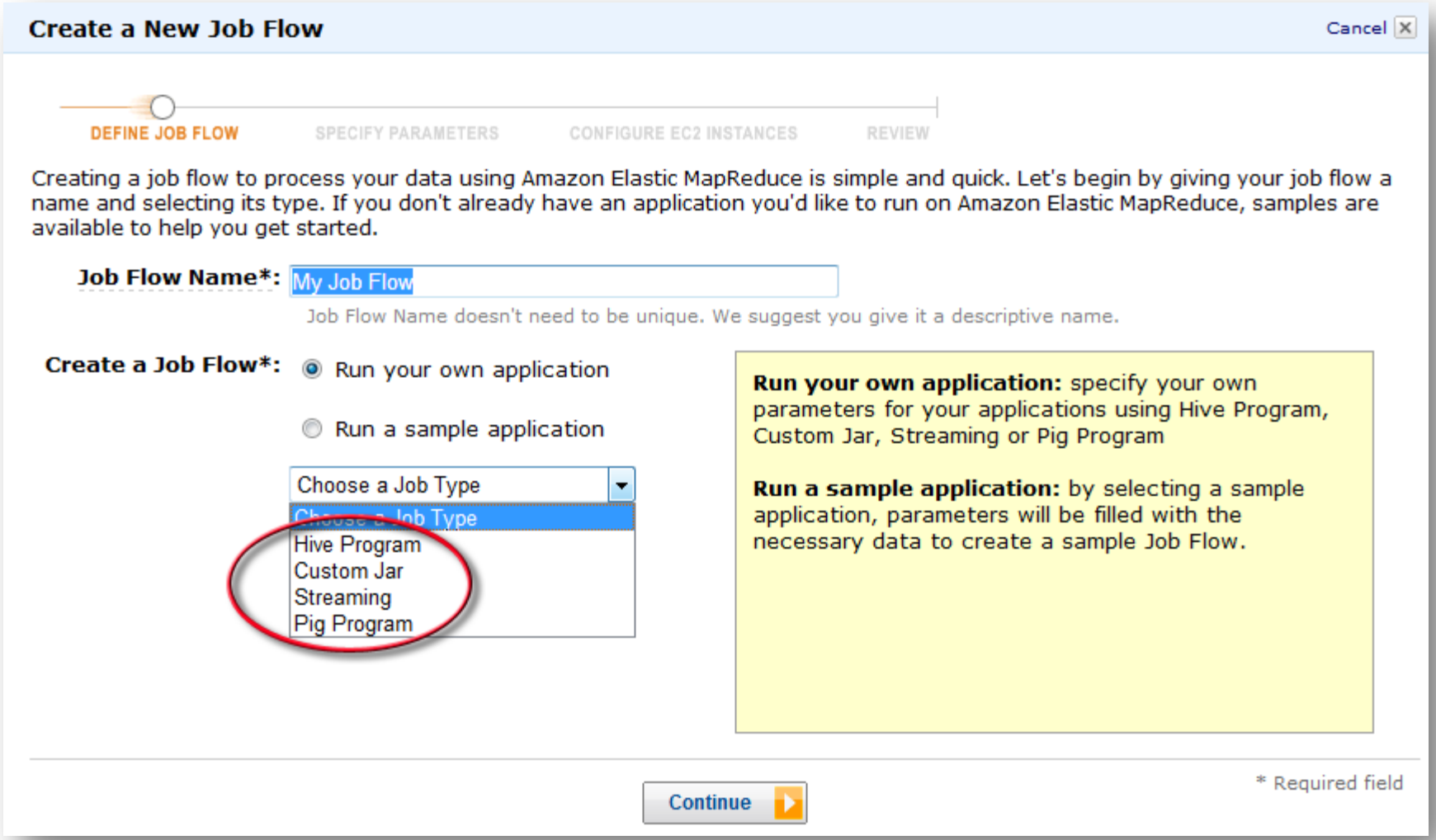


# Usability, Price & Performance

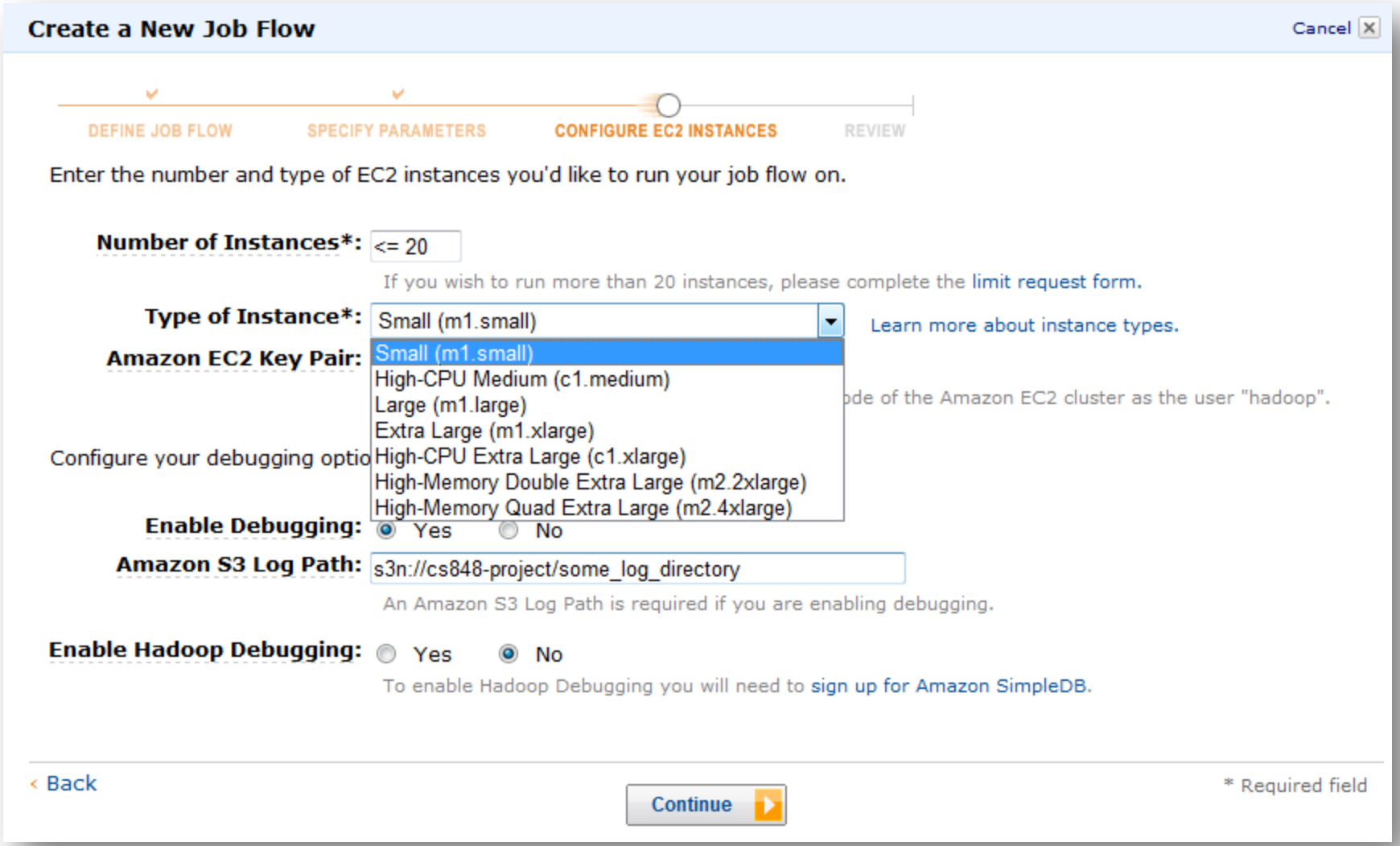
- Easy to use
- Large choice of programming languages
- Libraries for common programming languages
- But poor documentation and few examples
- Choice of different interfaces: web, command line or API
- Cheap enough to use as test environment
- High initialization times -> at least 10 minutes


Your Elastic MapReduce Job Flows
Region: US East
Create New Job Flow
Terminate
Debug
Show/Hide
Refresh
Help
Viewing: All
1 to 12 of 12 Job Flows
Name
State
Creation Date
Elapsed Time
Normalized Ins
PseudoRandomForceTest11Reduce
COMPLETED
2010-03-14 21:24 EDT
0 hours 4 minutes
2
PseudoRandomForceTest11Map
COMPLETED
2010-03-14 21:22 EDT
0 hours 4 minutes
2
PseudoRandomForceTest10
FAILED
2010-03-14 21:12 EDT
0 hours 2 minutes
2
PseudoRandomForceTest9
FAILED
2010-03-14 21:11 EDT
0 hours 2 minutes
2
PseudoRandomForceTest8
COMPLETED
2010-03-14 20:57 EDT
0 hours 3 minutes
2
1 Job Flow selected
Id: j-2941CDETYNWEZ
Name: PseudoRandomForceTest11Reduce
State: COMPLETED
Last State Change Reason: Steps completed
Availability Zone: us-east-1d
Master Type: m1.small
Key Name: cs848-project
Master Public DNS Name: ec2-174-129-55-147.compute-1.amazonaws.com
Creation Date: 2010-03-14 21:24 EDT
Start Date: 2010-03-14 21:29 EDT
End Date: 2010-03-14 21:33 EDT
Instance Count: 2
Slave Type: m1.small
Log URI: s3n://cs848-project/test11reduce.log/
Steps:
Step Name
State
Start Date
End Date
Jar
Main Class
Args
Custom Jar
COMPLETED
2010-03-14 21:29 EDT
2010-03-14 21:32 EDT
s3n://cs848-project/PseudoRondomForce.jar
PRF_Reducer -minK 1 -maxK 10000 -Y0 125 -len 20000 -o s3n://cs848-project/test11reduce -s 0408

# Amazon Elastic MapReduce



# Amazon Elastic MapReduce



# Our algorithm

- Is a **brute force** attack that checks all possible initial seeds
- It is given a short sequence of consecutively generated numbers as input
- **N** is a global program parameter representing the maximum length of a generated sequence
- The result is a list of pairs of initial seeds and positions within the generated sequence that match the input sequence

# Example

- N = 10, input sequence = {59, 82, 53}

| seed | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 129,881 | 6 | 74 | 59 | 82 | 53 | 42 | 78 | 62 | 20 | 39 |
| 190,847 | 94 | 65 | 76 | 40 | 78 | 95 | 12 | 69 | 46 | 83 |
| 213,900 | 49 | 62 | 52 | 45 | 17 | 49 | 59 | 82 | 53 | 39 |
| 428,605 | 68 | 68 | 38 | 68 | 94 | 27 | 74 | 15 | 59 | 91 |
| 543,902 | 43 | 59 | 82 | 53 | 90 | 45 | 12 | 44 | 13 | 65 |
| 873,593 | 1 | 68 | 42 | 73 | 94 | 52 | 26 | 74 | 67 | 67 |

- Output = ?

# Example

- N = 10, input sequence = {59, 82, 53}

| seed | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 129,881 | 6 | 74 | 59 | 82 | 53 | 42 | 78 | 62 | 20 | 39 |
| 190,847 | 94 | 65 | 76 | 40 | 78 | 95 | 12 | 69 | 46 | 83 |
| 213,900 | 49 | 62 | 52 | 45 | 17 | 49 | 59 | 82 | 53 | 39 |
| 428,605 | 68 | 68 | 38 | 68 | 94 | 27 | 74 | 15 | 59 | 91 |
| 543,902 | 43 | 59 | 82 | 53 | 90 | 45 | 12 | 44 | 13 | 65 |
| 873,593 | 1 | 68 | 42 | 73 | 94 | 52 | 26 | 74 | 67 | 67 |

- Output = {?, ?, ?}

# Example

- N = 10, input sequence = {59, 82, 53}

| seed | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 129,881 | 6 | 74 | 59 | 82 | 53 | 42 | 78 | 62 | 20 | 39 |
| 190,847 | 94 | 65 | 76 | 40 | 78 | 95 | 12 | 69 | 46 | 83 |
| 213,900 | 49 | 62 | 52 | 45 | 17 | 49 | 59 | 82 | 53 | 39 |
| 428,605 | 68 | 68 | 38 | 68 | 94 | 27 | 74 | 15 | 59 | 91 |
| 543,902 | 43 | 59 | 82 | 53 | 90 | 45 | 12 | 44 | 13 | 65 |
| 873,593 | 1 | 68 | 42 | 73 | 94 | 52 | 26 | 74 | 67 | 67 |

- Output = {(129881, 4), ?, ?}

# Example

- N = 10, input sequence = {59, 82, 53}

| seed | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 129,881 | 6 | 74 | 59 | 82 | 53 | 42 | 78 | 62 | 20 | 39 |
| 190,847 | 94 | 65 | 76 | 40 | 78 | 95 | 12 | 69 | 46 | 83 |
| 213,900 | 49 | 62 | 52 | 45 | 17 | 49 | 59 | 82 | 53 | 39 |
| 428,605 | 68 | 68 | 38 | 68 | 94 | 27 | 74 | 15 | 59 | 91 |
| 543,902 | 43 | 59 | 82 | 53 | 90 | 45 | 12 | 44 | 13 | 65 |
| 873,593 | 1 | 68 | 42 | 73 | 94 | 52 | 26 | 74 | 67 | 67 |

- Output = {(129881, 4), (213900, 8), ?}

# Example

- N = 10, input sequence = {59, 82, 53}

| seed | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 129,881 | 6 | 74 | 59 | 82 | 53 | 42 | 78 | 62 | 20 | 39 |
| 190,847 | 94 | 65 | 76 | 40 | 78 | 95 | 12 | 69 | 46 | 83 |
| 213,900 | 49 | 62 | 52 | 45 | 17 | 49 | 59 | 82 | 53 | 39 |
| 428,605 | 68 | 68 | 38 | 68 | 94 | 27 | 74 | 15 | 59 | 91 |
| 543,902 | 43 | 59 | 82 | 53 | 90 | 45 | 12 | 44 | 13 | 65 |
| 873,593 | 1 | 68 | 42 | 73 | 94 | 52 | 26 | 74 | 67 | 67 |

- Output = {(129881, 4), (213900, 8), (543902, 3)}

# Experiments

- MapReduce scalability (up to 20 EC2 instances)
- Different hardware setups (small/large EC2 instances)
- Day versus Night job completion time
- Different instance regions' job completion time
- Calculations performed in the mapping stage versus the reduction stage
- Generated input data versus data read from S3

# Conclusion

- This is an experiment project studying seed identification attacks on pseudo-random number generators
- Amazon Elastic MapReduce is a simple and fast way to deploy MapReduce operations
- It's both easy and cheap to get started with Amazon Elastic MapReduce and later scale out without ever investing in an infrastructure
- Experiments will reveal the system's performance and more on its financial costs